(12) UK Patent Application (19) GB (11) 2 367 661 (13) A

(43) Date of A Publication 10.04.2002

(21) Application No 0104912.1

(22) Date of Filing 28.02.2001

(30) Priority Data
(31) 09522416 (32) 09.03.2000 (33) US

(71) Applicant(s)
International Business Machines Corporation
(Incorporated in USA - New York)
Armonk, New York 10504, United States of America

(72) Inventor(s)
Amir Herzberg
Yiftach Ravid

(74) Agent and/or Address for Service
S R Davies
IBM United Kingdom Limited, Intellectual Property
Dept, Hursley Park, WINCHESTER, Hampshire,
SO21 2JN, United Kingdom

(51) INT CL^7
G06F 17/30

(52) UK CL (Edition T)
G4A AUDB

(56) Documents Cited
WO 00/55741 A1

(58) Field of Search
UK CL (Edition T) G4A AKS AUDB
INT CL^7 G06F 3/00 17/30
Online: EPODOC, WPI, Japio

(54) Abstract Title
Managing objects

(57) A method for managing objects for users includes providing a set of attributes and a set of containers each having attributes from the set. A user interface is provided for dynamically assigning attributes to the objects, and for selectively displaying containers and objects in the containers. An object is displayed in a container if a condition is met, the condition being applied to the attributes of the container and the attributes of the object. A computer system for managing the objects is also disclosed. Utility can be in sharing of bookmarks of frequently used URLs among web surfers.

GB 2 367 661 A

TREE 1 (222 x 147 x 16M jpeg)

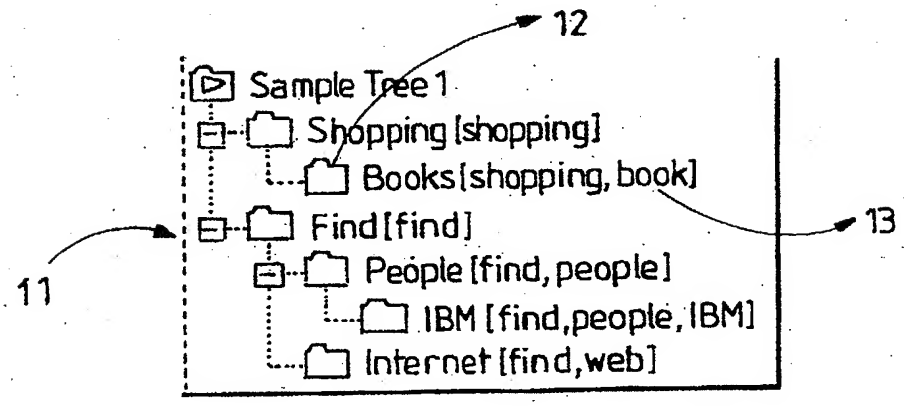


FIG. 1A

TREE 2 (222 x 164 x 16M jpeg)

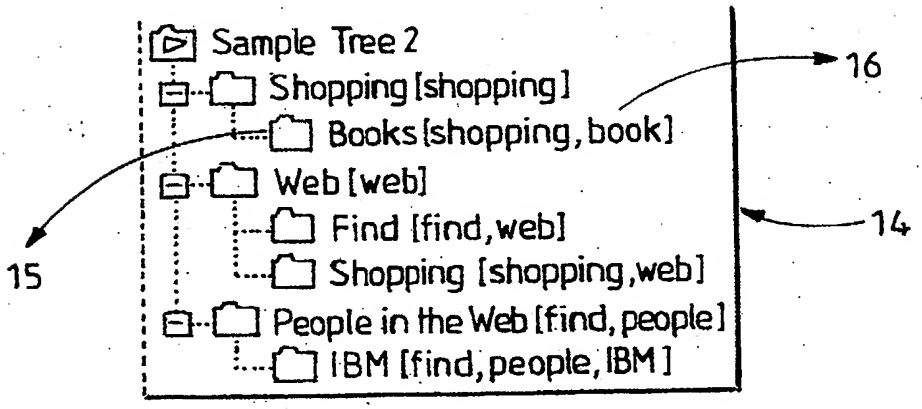


FIG. 1B

(399 x 399 x 16M jpeg)

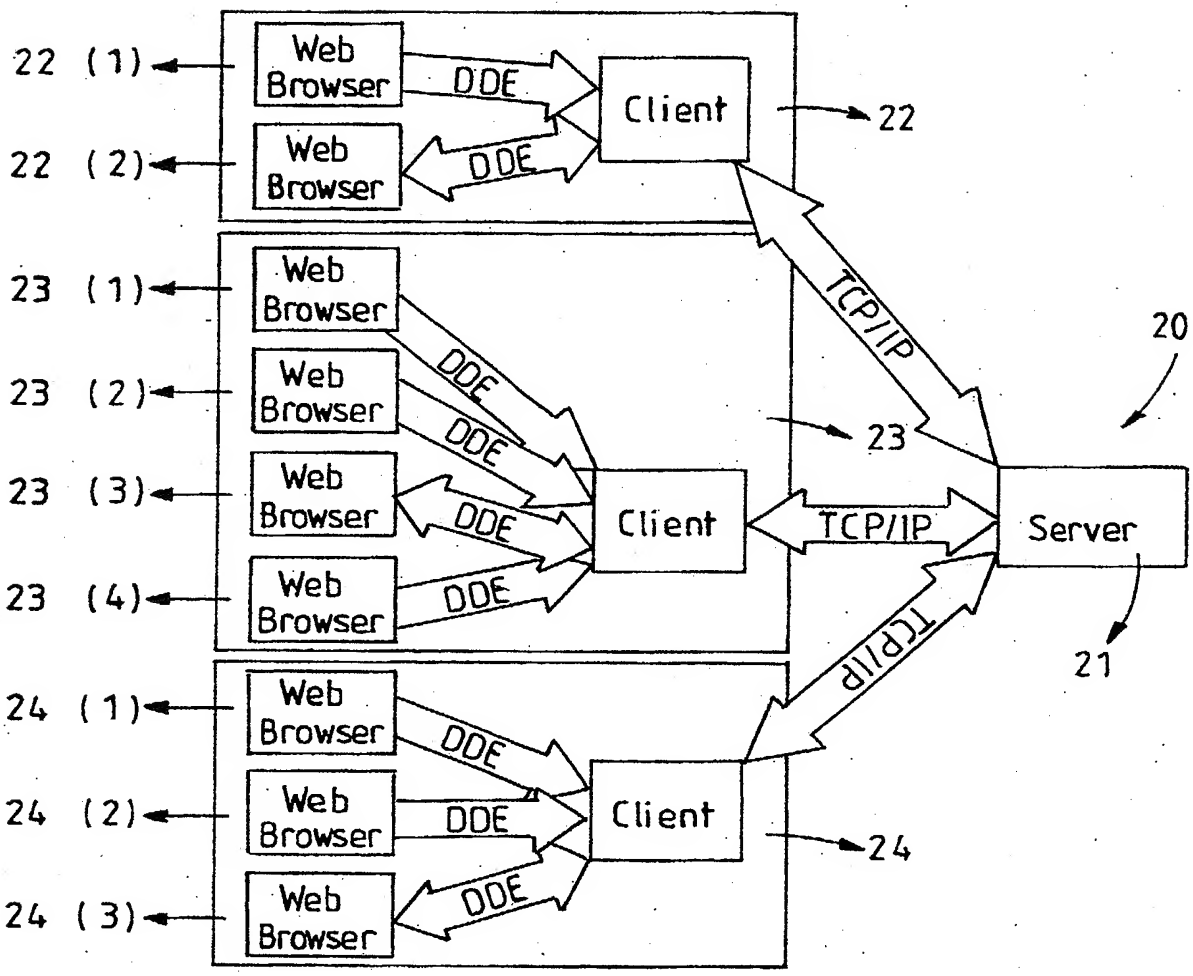


FIG. 2

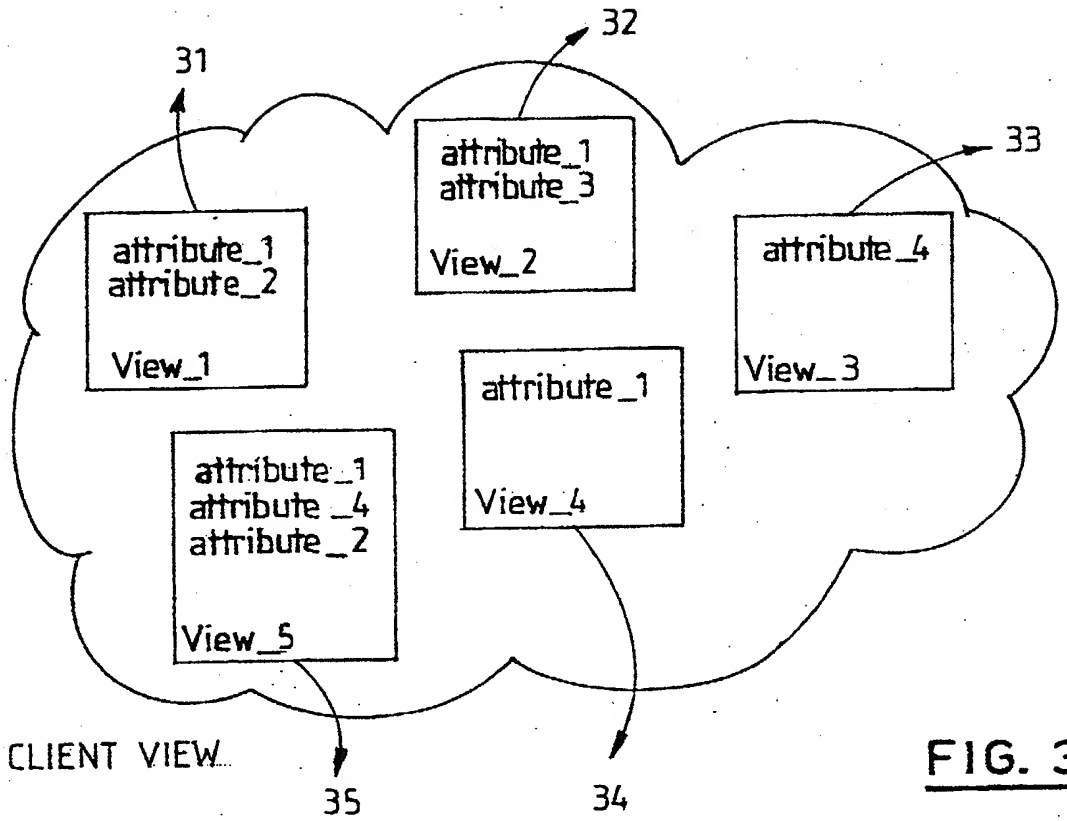


FIG. 3A

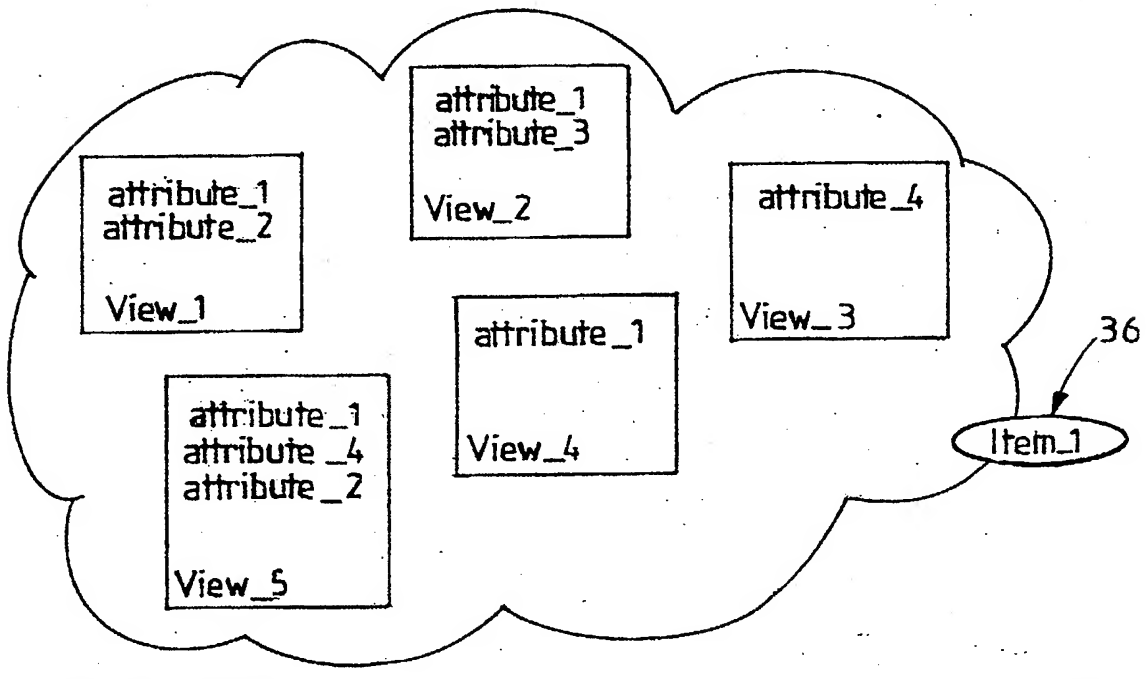


FIG. 3B

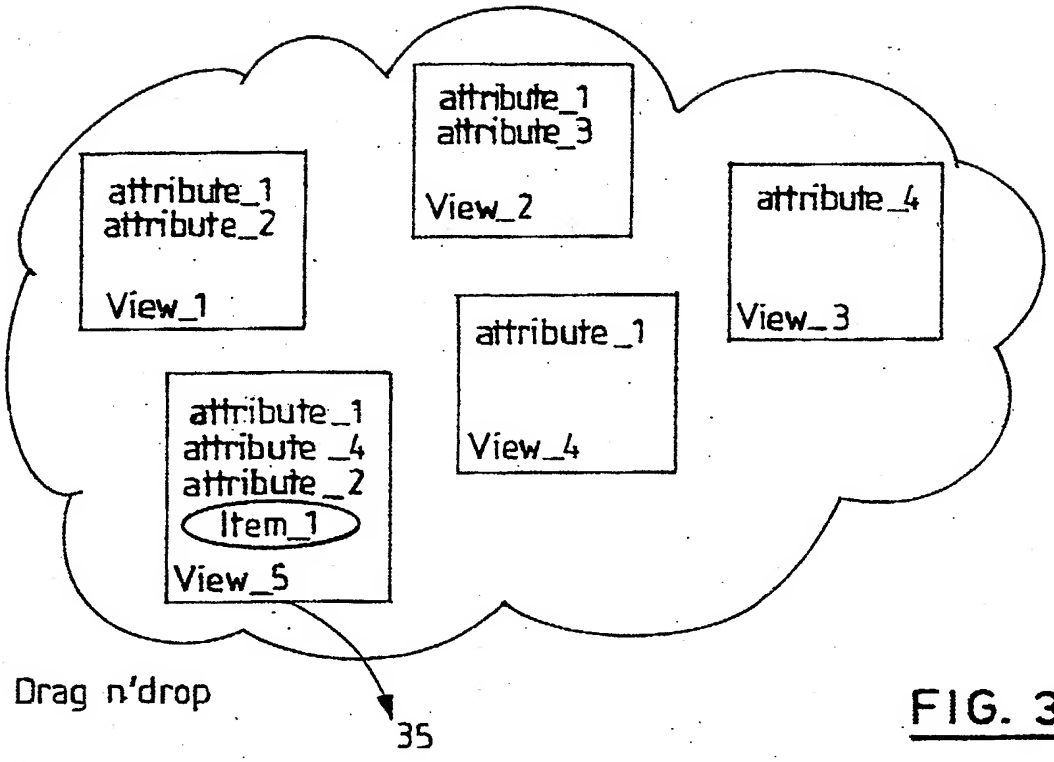


FIG. 3C

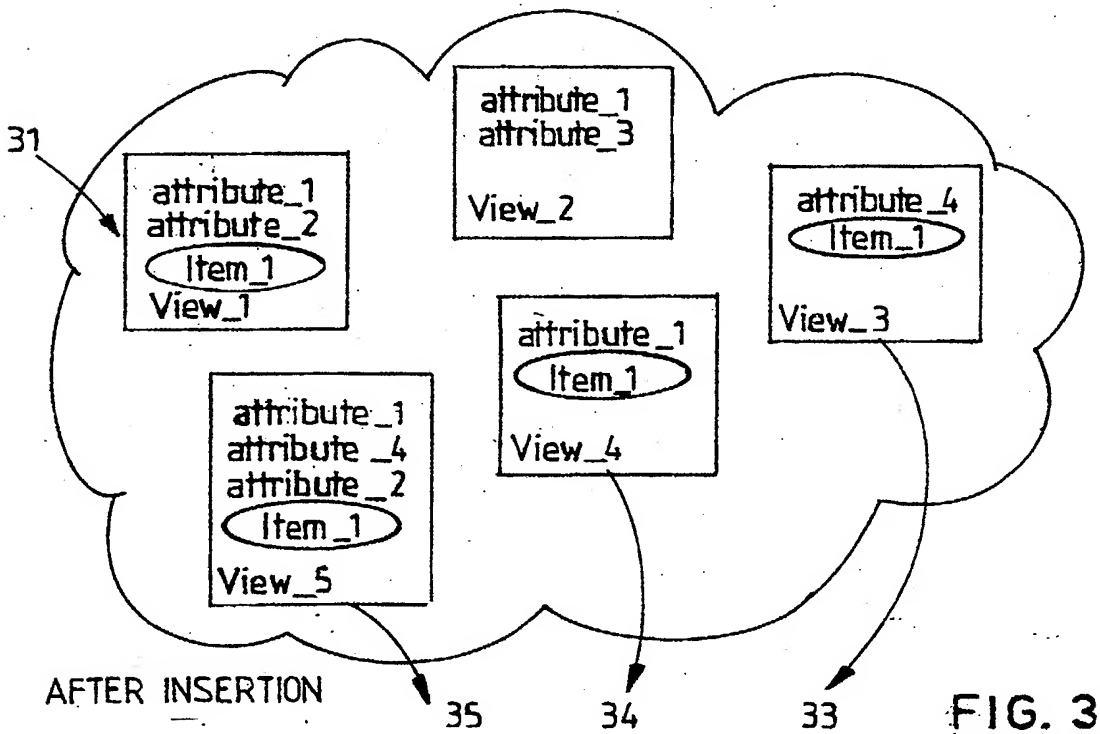


FIG. 3D

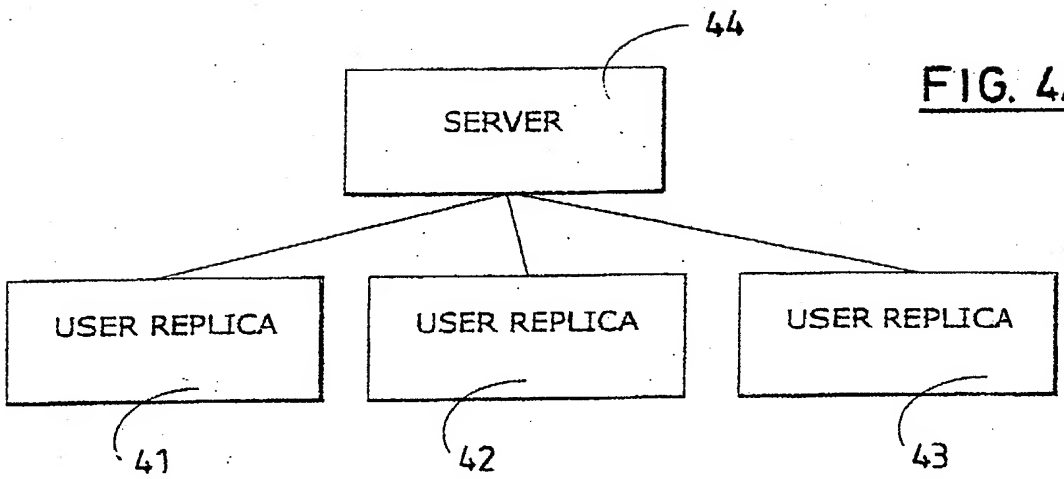


FIG. 4A

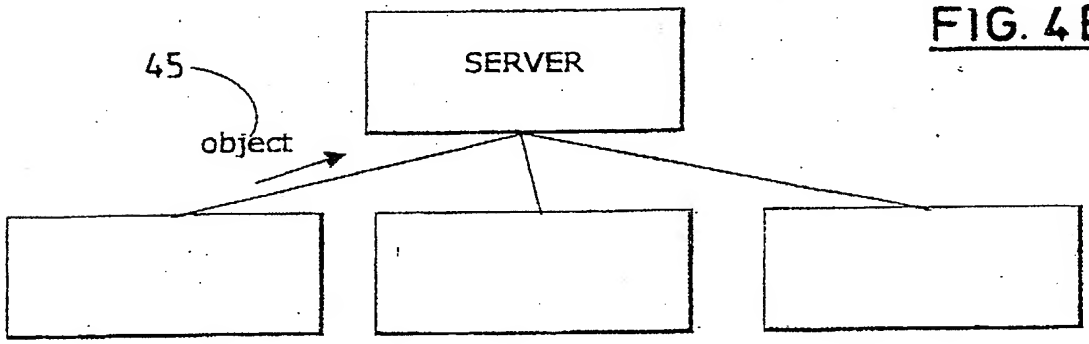


FIG. 4B

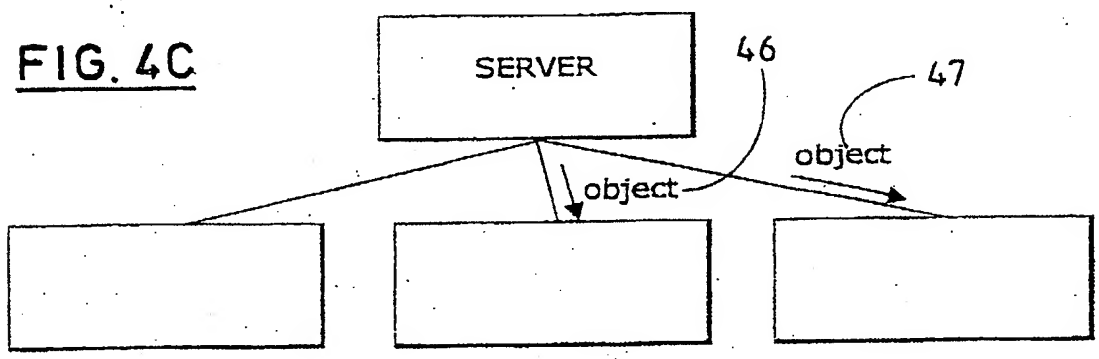


FIG. 4C

FIG. 4

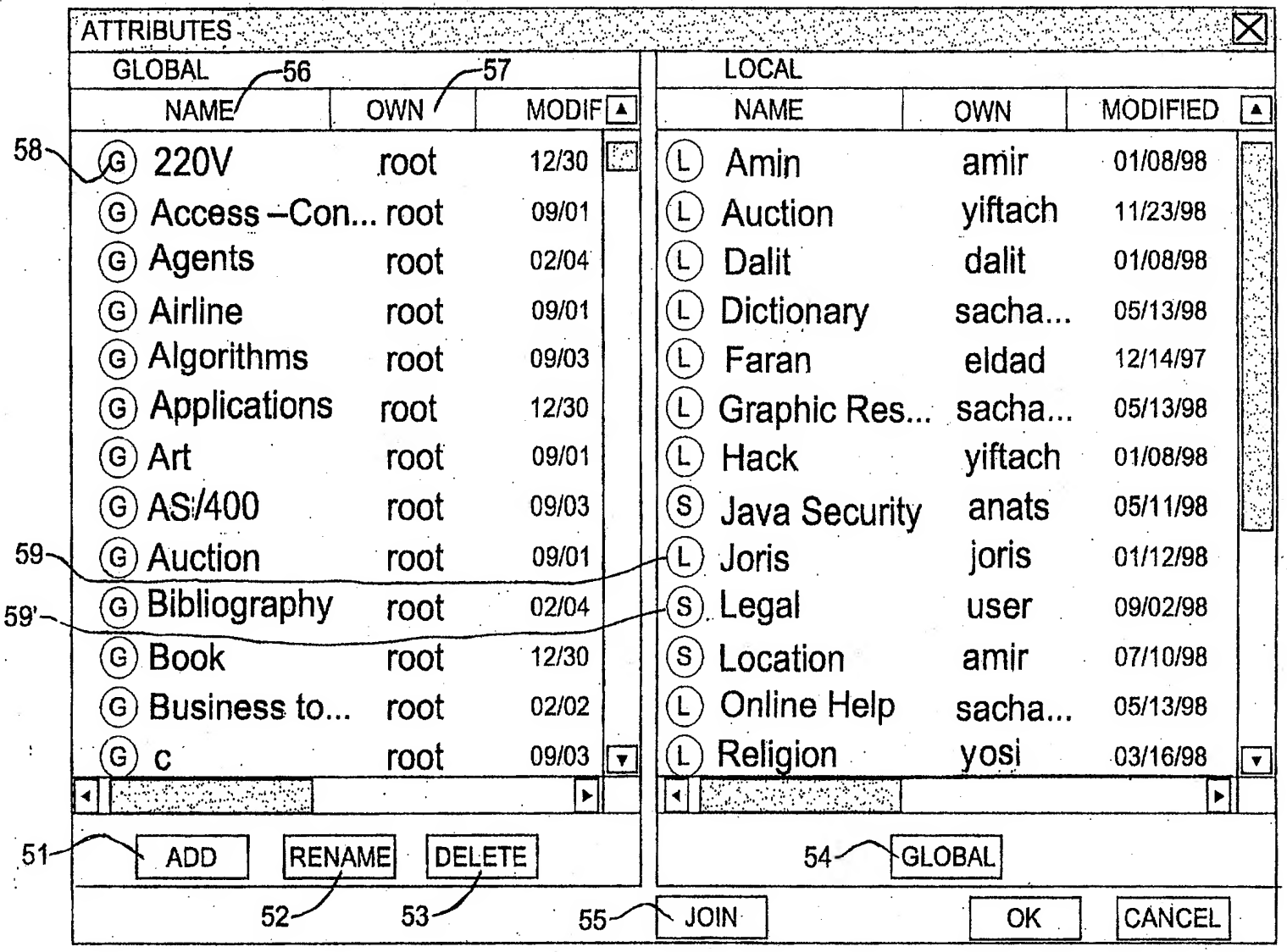


FIG. 5

(474 x 409 x 16M jpeg)

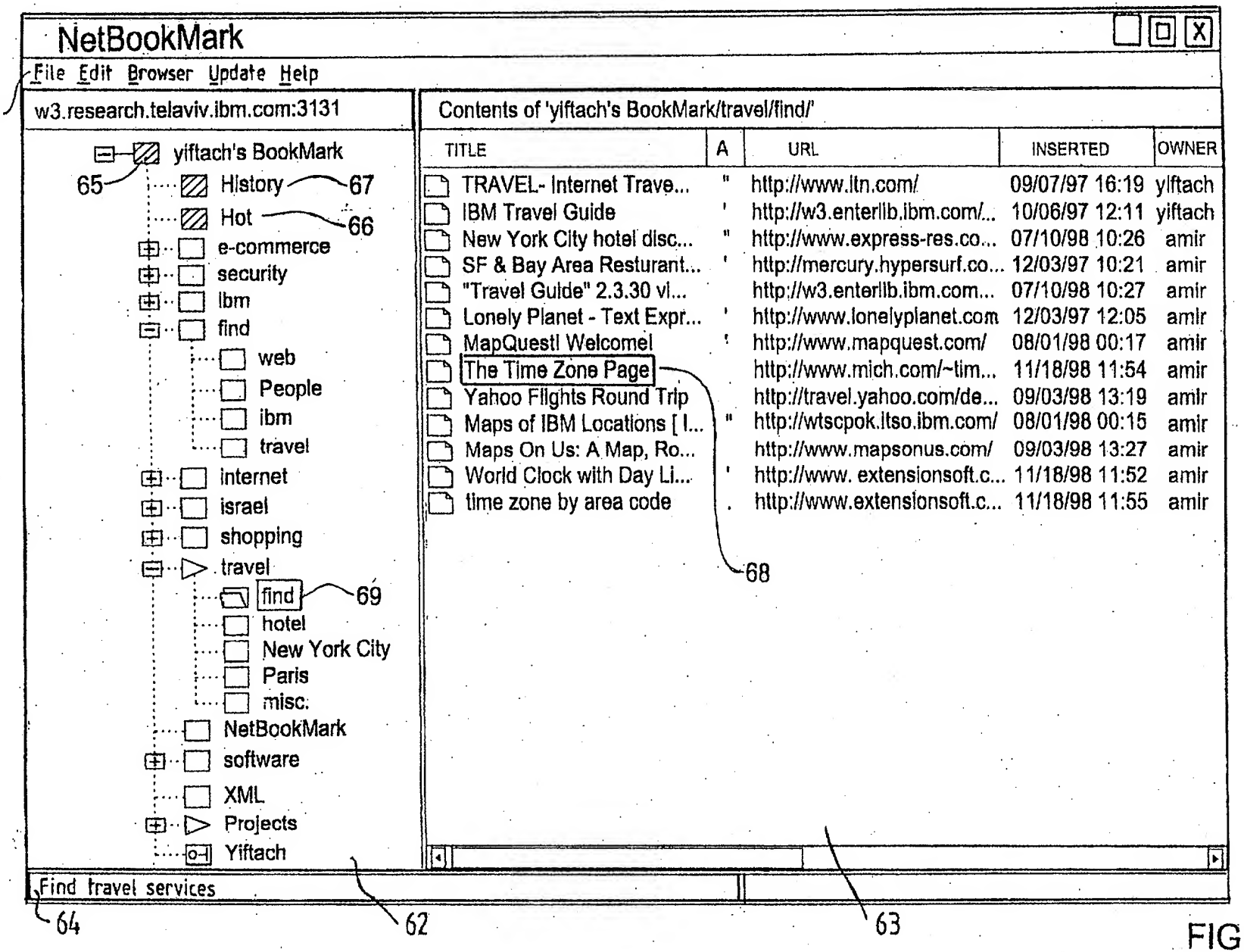
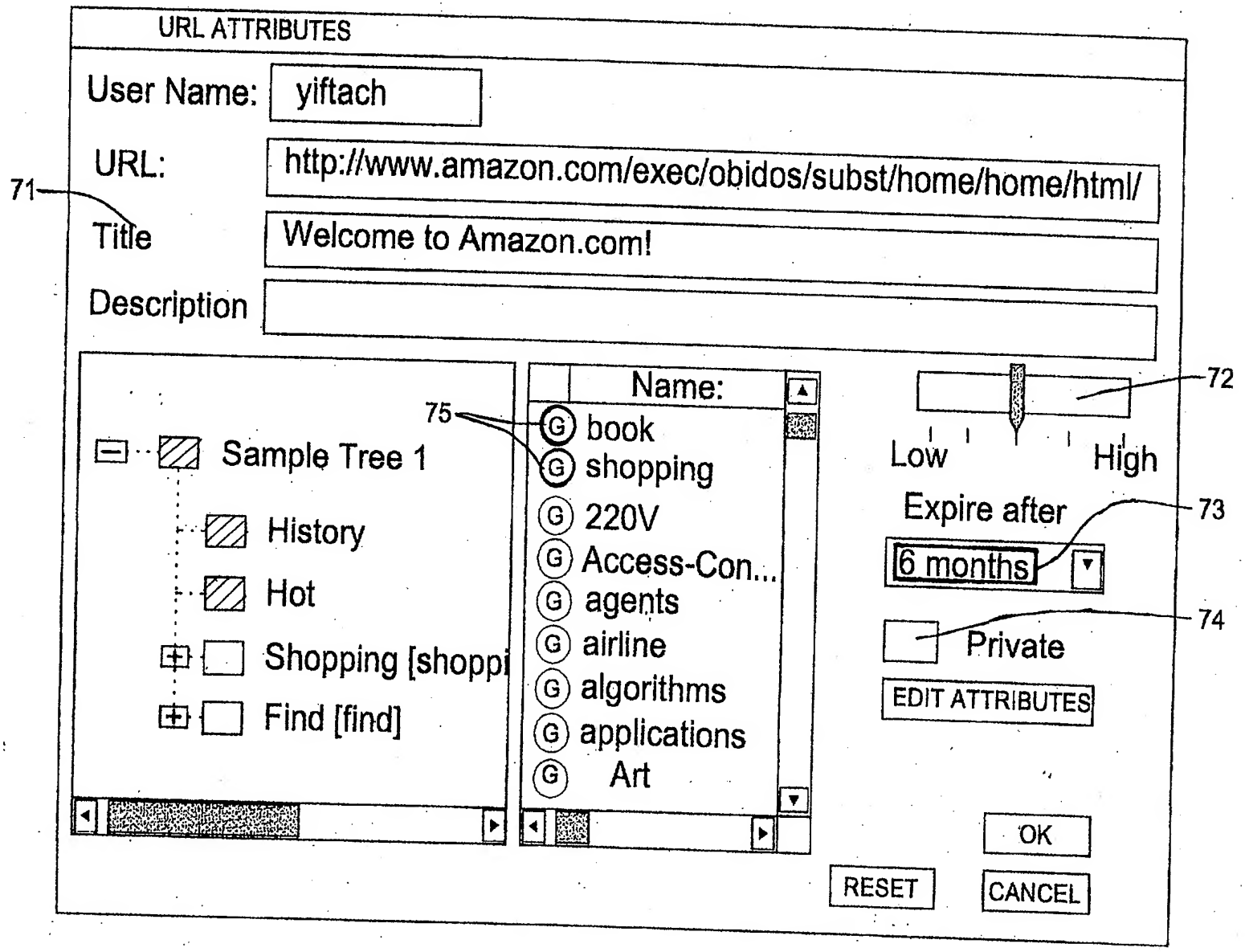


FIG. 6

(768 x 547 x 16M jpeg)



(395 x 379 x 16M jpeg)

FIG. 7

A METHOD AND SYSTEM FOR MANAGING OBJECTS

The present invention relates generally to managing objects and sharing information among communities, such as sharing bookmarks of frequently used locations (URLs) among web surfers.

The following documents are indicative of the prior art in this field and are discussed in more detail below:

(a) Keller R.M., Wolfe S.R., Chen J.R., Rabinowitz J.L. and Mathe N., "A Bookmark Service for Organizing and Sharing URLs", *sixthinternational World Wide Web Conference,* Santa Clara, California, USA, April 7-11, 1997.

(b) Klark, P. and Manber, U. "Developing a Personal Internet Assistant", *Proc. Conference on Educational Multimedia and Hypermedia (ED-MEDIA '95)*Graz, 372-377, 1995.

(c) Wittenberg, K. Das, D. Hill, W. and Stead, L., "Group Asynchronous Browsing on the World Wide Web" *Fourth Additional World Wide Web Conference,* Boston, December 11-14, 1996.

(d) Goldberg, D., Nichols, D, Oki, B.M. and Terry, D., "Using Collaborative Filtering to Weave an Information Tapestry" *Communications of the ACM* 35 (12), 61-70, 1992.

(e) http://www.imana.com/WebObjects/Siteseer

(f) Kamiya,K., Rvscheisen, M. and Winograd, T., "Grassroots: A System Providing Uniform Framework for Communicating, Structuring, Sharing Information and Organizing People", *Fifth International World Wide Web Conference* Paris, France, 6-10, 1996.

(g) Konstan, J. Miller, B., Maltz, D., Herlocker, J., Gordon, L. and Riedl, J., "GroupLens: Applying Collaborative Filtering to Usenet News".*Communications of the ACM 40,3* (1997), 77-87.

(h) David K. Gifford, Pierre Jouvelot, Mark A. Sheldon, James W. O'Toole, Jr., "Semantic File Systems" *Programming Systems Research Group* - MIT Laboratory for Computer Science.

(i) Burra Gopal, Udi Manber, "Integrating Content-Based Access Mechanisms with Hierarchical File Systems", *Third Symposium on Operating Systems Design and Implementation,* (OSDI '99) Usenix Association.

The Web is huge. Users often find locations (sites, home pages, URLs) which they feel are of potential future use. Finding their way to the same location again can be tricky, difficult and time consuming. Accordingly, surfers usually want to mark a web page of interest, making it easier to find when necessary. These marks are usually referred to as *bookmarks* (also as favorites, thumbnails, hotlists etc).

The conventional way to organize bookmarks is using a tree structure. Popular browsers (Netscape Communicator, Netscape Navigator and Microsoft Internet Explorer; trademarks of their respective companies) include a bookmark management tool, each using its own proprietary format for storing the bookmarks. Several stand-alone bookmark management tools have also been developed These tools allow users to design their own tree organization of bookmarks, and to enter bookmarks and manage them, as well as open them in the browser (when clicked). The tree organization is a simple and well understood way, allowing each user to organize bookmarks for easy access based on personal preferences.

These existing bookmark management tools, with a few exceptions noted later, are focused on helping an individual user rather than facilitating the sharing of bookmarks among users. In order to share bookmarks, users typically send the URL's or bookmark files to each other (e.g. using e-mail). This is clearly not a very efficient mechanism. Sharing of complete bookmark files is especially problematic as different tools (browsers) use different proprietary, incompatible formats. This is actually a problem even for users who use multiple browsers and/or machines - e.g. users that have two operating systems or computers, and two different browsers will have four bookmark files. There are some utilities that purport to translate bookmark files among these different formats, but this procedure is burdensome and sometimes error prone.

Another common approach used to share URLs is to provide them as links on specially designed web pages, referred to as *web indexes* (or "lists of links"). Such indexes are now commonly provided at most web sites and even on many personal homepages. Indexes may focus on the interests of a particular individual or community, on a particular subject, or be general. There are several large, mega-indexes which categorize "all the web" as a series of HTML pages. A well known example is YAHOO! Each index has its own tree structure, and typically offers two ways of locating URLs (i.e. bookmarks); using a search utility (simple search in a database), or using the categorized folders. In the mega-indexes, users may also mark web pages to be indexed and suggest the appropriate folder.

There are several problems in sharing bookmarks using such indexes:

(a) The indexes are provided in special web sites, rather than as a local application and files; this implies that access is substantially slower and less convenient, compared to a local bookmark management tool. Furthermore, this depends on connectivity to the index server.

(b) The same URL may be relevant to more than one folder; furthermore, different users may prefer different arrangements of folders (e.g. top level folder "music" and sub-folder "shopping" or vice versa) or simply a

different name for folders (e.g. "find" and "search"). The result is that relevant URLs are often spread cross multiple folders, as is known to every user of popular indexes such as Yahoo!

(c) Users often have bookmarks that they do not wish to share, being private (e.g. including passwords) or simply user specific. A common index server does not allow such features.

Several different systems enable users to share personal bookmark collections (references are to the full citations given above):

Keller *et al.* describe the WebTagger personal and community bookmarking service, running in a web server (as a CGI program). The system is a proxy based system that modifies each web page being browsed, by adding buttons for categorizing the page, querying the database, etc. To look up folders, the user needs to form a query, as there is no common or customized folder organization. *Inter alia*, there is no support for privacy or replication.

Klark and Manber designed the Warmlist. The application is based on sharing a common bookmark file using a shared file system (Unix). Users can insert their own bookmarks, organize them hierarchically (using the same tree), index them, and search the file for bookmarks.

Active Notebook allows users to label information with conceptual classifications and organize them into a taxonomy for later browsing and retrieval. The focus of this work is on clustering documents and identifying morphological concepts (keywords); it does not deal with aspects of sharing such as replication, privacy and user interface.

Wittenberg *et al.* created the Group Asynchronous Browsing Server that collects and merges user's bookmarks and then displays these merged bookmarks in a standard web browser. Their approach is that when they find the same URL in two bookmark files, they give the user a link from the folder in one bookmark file to the folder of the other bookmark file.

Collaborative filtering methods provide a means of selective information sharing by utilizing preference indicated by other users. These preferences might be inferred implicitly from the actions of others, or might be based on explicit user evaluation. See also the above references to Goldberg *et al.*, to SiteSeer, to Kamiya *et al.*, and to Konstan *et al.*.

Gifford *et al.* provides for a *semantic file system* which is an information storage system that provides flexible associative access to the system's contents by automatically extracting attributes from files with

file type specific *transducers*. Associative access is provided by a conservative extension to existing tree-structured file system protocols, and by protocols that are designed specifically for content based access. Compatibility with existing file system protocols is provided by introducing the concept of a *virtual directory*. Virtual directory names are interpreted as queries, and thus provide flexible associative access to files and directories in a manner compatible with existing software. Rapid attribute-based access to file system contents is implemented by automatic extraction and indexing of key properties of file system objects. The Semantic File System in accordance with the Gifford *et al.* publication supports various types of objects, such as documents, emails and other objects.

Gopal *et al.* provides for a new file system that combines name-based and content-based access to files at the same time. The design allows both methods to be used at any time, thus preserving the benefits of both. Users can create their own name spaces based on queries, on explicit path names, or on any combination interleaved arbitrarily. All regular file operations – such as adding, deleting, or moving files- are supported in the same way, and in addition, query consistency is maintained and adapted to what the user is manually doing. One can add, remove or move results of queries, and in general handle them as if they were regular files.

The specified Gifford *et al.* and Gopal *et al.* publications offer an object management system which does not aim at handling object sharing, but which rather supports handling of, say, data files, by assigning attributes to the specified data files, and enabling a user to query the data files by the attributes - for example, by extracting all data files sent *to "Smith"*. This approach may support also *bookmark* management in a similar manner.

Systems of the kind disclosed in Gifford *et al.* and Gopal *et al.* publications, have some inherent limitations. For one, the attributes that are assigned to the various objects are static in nature. (e.g. for letters the fields: *From:*, *To:* etc.). In many real life applications the static nature of attributes is insufficient. Consider, for example, the specified bookmark sharing application where a given bookmark is assigned with the attribute, say *IBM*. A given user may now decide that the specified bookmark should also be assigned with the attribute, say *security*. Not only does a static attribute system fall short in supporting such change, but obviously also fails to provide a scheme for propagating this update among other members in the community. Thus, it is desired that the specified bookmark is reflected at other user views (associated with the same or different users) who already exploit the attribute *security*.

Accordingly, the invention provides a method for managing objects for at least one user, comprising:

providing a set of attributes;

providing a set of containers, each associated with one or more attributes from said set;

providing a user interface for dynamically assigning attributes from said set to said objects; and

selectively displaying, through a user interface, at least one container, wherein an object is displayed in said container if a predetermined condition is met, said condition involving at least one attribute of the container and at least one attribute of the object.

In a preferred embodiment, the dynamically assigning is performed by mapping an object, typically via a drag-and-drop operation, to a container, with the object then inheriting the attributes of the container.

In one embodiment, a container can be associated with one or more essential attributes. The condition for selective display then includes the requirement that the attributes of said objects contain the one or more essential attributes.

A preferred setting for the invention is where there is a community of users that share said objects. Each user has a respective set of attributes, at least one attribute being common to at least two of said users, and performs the following steps:

providing a user replica that includes objects that are each assigned one or more attributes;

providing a set of containers each associated with one or more attributes from said set;

providing a user interface for generating an update to said replica;

submitting the update stipulated to the replicas of selected other users;

receiving at least one update from another user in the community;

updating said user replica in accordance with the at least one received update; and

selectively displaying, through a user interface, at least one container, wherein an object from the replica is displayed in said container if a predetermined condition is met, said condition involving at least one attribute of the container and at least one attribute of the object.

In a preferred embodiment, objects can be categorised as private, and encrypted with a user unique key. Relatedly, one or more selected

containers in a user replica can be assigned a *private* attribute, whereby any object assigned to such a container is encrypted using a unique key.

In this embodiment, the set of attributes associated with each user forms part of the user replica, and an update includes assigning attributes to said objects, as well as updating an attribute in said set, by adding an attribute, deleting an attribute, editing an attribute, change the status of attribute from Global to Local or *vice versa*, and joining two or more attributes into a single attribute (or any other such action).

In a preferred embodiment, the user interface includes a tree of containers, the containers being folders. An object is displayed in a container only if it is not displayed in a sub-container thereof. The objects typically represent URL bookmarks of web sites, or files.

The invention further provides a computer program comprising a set of program instructions which when followed on one or more computer systems cause said one or more computer systems to perform a method of the invention. Typically the computer program will be supplied either as software for download over a network, recorded onto a storage medium such as a DVD or CD ROM, or preloaded into a computer system.

The invention further provides a computer system for managing objects for at least one user, comprising:

means for providing a set of attributes;

means for providing a set of containers, each associated with one or more attributes from said set;

means for providing a user interface for dynamically assigning attributes from said set to said objects; and

means for selectively displaying, through a user interface, at least one container, wherein an object is displayed in said container if a predetermined condition is met, said condition involving at least one attribute of the container and at least one attribute of the object.

It will be appreciated that the computer program and computer system of the present invention will benefit from the same preferred features as the method of the invention.

Thus the approach described above allows objects to be managed either in a standalone environment, or, if desired, shared among users, overcoming some of the deficiencies of the prior art.

One particularly preferred embodiment of the invention provides a bookmark sharing application including the following characteristics:

Common attributes: the community all uses the same set of attributes. Each user builds his/her own user interface, in this particular example a tree of folders, associating attributes to each folder. Each bookmark is also assigned attributes which are used to display the bookmark in the appropriate folders for each user (if a condition is met). The list of attributes can be updated, e.g. new attributes can be inserted and/or others may be deleted.

Replication: a database of all bookmarks is kept in a server to which all users have access. Preferably, local replicas of this database are kept in each user's machine. The replication enables users *inter alia* to work off-line (while not connected to a server), and provides much better performance.

Privacy: users may easily define some URLs (and potentially attributes) as private. Privately marked URLs are encrypted in the server and in the replicas of that user, so that access is possible only using the key of the user. *Privacy* may be regarded as an attribute that is associated with selected folders. Bookmarks can be dynamically assigned to folders and in this preferred embodiment they inherent the attributes of the folder. Thus bookmark that are assigned to *private* folders inherit the privacy attribute.

Other attributes may be assigned to folders such as, for example, *History* signifying that bookmarks associated with this folder have been recently used. Another example is a *Hot* attribute associated with a folder that includes all those bookmarks that have been recently added to folders.

Simple, familiar view user interface: the implementation uses a familiar tree folder structure user interface, allowing the user to perform common operations with a drag and drop user interface. The approach resembles common known user interfaces, such, for example, in the MICROSOFT folder Explorer utility, and therefore reduces the time it takes to become familiar with the user interface. The *drag and drop* operation is utilized in one embodiment to assign attributes to bookmarks. Having mapped the bookmark to a folder the attributes of the folder are automatically assigned to the bookmark.

Display of bookmarks in folders: as will be explained in greater detail below, the assignment of attributes to bookmarks, e.g. by drag and dropping the specified bookmark to a folder, does not necessarily mean that the bookmark will be displayed in the specified folder. In order for the bookmark to be displayed in the folder, it must meet a condition(s) that is (are) applied to the attributes of the container and the bookmark.

A preferred embodiment of the invention will now be described in detail by way of example only with reference to the following drawings:

Figs. 1a-b illustrate a sample user interface view represented as a tree of folders, in accordance with one embodiment of a bookmark sharing application;

Fig. 2 illustrates a block diagram of a generalized system architecture in accordance with a preferred embodiment of the invention;

Figs. 3a-d illustrate a user view before and after the insertion of an object;

Figs. 4a-c illustrate an update sequence of operation in accordance with one embodiment of the invention;

Fig. 5 illustrates an exemplary attributes dialog box, in accordance with on embodiment of the invention;

Fig. 6 illustrates a user interface view in accordance with one embodiment of the invention; and

Fig. 7 illustrates a bookmark dialog box in accordance with one embodiment of the invention.

A popular interface for viewing bookmarks (as well as files, mail messages, etc.), is using a tree of folders, e.g. in accordance with the Microsoft file explorer user interface. In many applications, browsing the tree is a better way to look for the right bookmark, rather than doing a textual search in the database. However, organizing shared bookmarks into folders is difficult. The same URL may be relevant to more than one folder; furthermore, different users may prefer a different arrangement of folders (e.g., one user prefers top level folder "music" and sub-folder "shopping", whereas another user prefers this the other way around, i.e. top level folder "shopping" and sub-folder "music"), or simply a different name for the folders (e.g., "find" and "search").

A simple approach that purports to overcome this problem is to decide on a fixed tree of folders used by all members of a community when entering bookmarks, possibly with some additional conventions. However, as explained above, such an approach lacks flexibility, and it is hard to agree on a commonly accepted and useful organization.

In a preferred embodiment of the invention, there is a community of users that agree on a set of *attributes*. Each user can build his/her set of containers and associate attributes with each container. It should be noted that whilst, preferably, the set of attributes is common to the users in the community, this is not necessarily always the case. Thus, users in the community may have a set of attributes, some of which are not necessarily shared by some or all of the other members in the community. A user interface is provided for dynamically adding, deleting or updating

attributes in the set. As will be explained in greater detail below, the attributes are used to display bookmarks in the appropriate folders of each user.

In the preferred embodiment, a user defines a set of attributes. In the case of a community of users, each user in a community defines a set of attributes that is relevant to the community's interests. The set of attributes is finite and yet dynamic. The procedure for addition of new attributes is explained in further below. In the case of community of users this set of attributes may be regarded as the "language" of the community.

The set of attributes is used to display bookmarks in, say, a tree of folders. A bookmark is displayed in a folder (or folders) if a predetermined condition is met, the condition applying to at least one of the attributes of the folder and at least one of the attribute of the bookmark. In one preferred embodiment, a bookmark is displayed in a folder if the bookmark and the folder share a common attribute.

To this end, objects (bookmarks) are dynamically assigned with attributes through a user interface. In a specific embodiment the attributes are dynamically assigned to bookmarks using a "drag and drop" operation. In this embodiment an object is mapped to a given container (folder) and inherits the attributes that are assigned to the specified container (folder).

For a better understanding, attention is now directed to Figs. 1A-B, illustrating a sample user interface represented as a tree of folders. Thus, the user interface 11 in Fig. 1A is represented as a tree of folders (for, say, a first user in the community, or a single user in a stand-alone environment) similar to the known Microsoft Explorer tree representation for files. Each folder is associated with attributes; in the example of Fig. 1A, the folder "books" (12) is associated with the attributes "shopping" and "book"(13). Note that in the user interface of Fig. 1A, attributes are embraced by square brackets. Sample folder tree 14 in Fig. 1B represents the user interface for, say, a second user in the community.

The list of URLs (or bookmarks) that corresponds to a given folder, are stored in the specified folder (not shown in Figs. 1A and 1B), similar to files that belong to a given folder in the Windows Explorer utility.

The table below lists some examples of where bookmarks with specific attributes will be placed with reference to Figs. 1A and 1B in accordance with one set of placement rules:

Attributes	Sample Tree 1	Sample Tree 2
Shopping	Shopping	Shopping
Shopping, web	Shopping	Web>Shopping, Shopping
Shopping, web, find	Shopping, Find>Internet	Shopping, Web>Shopping Web>Find
Find, people	Find>People	People in the Web
Find	Find	(Will not appear)

In one preferred embodiment of bookmark sharing, when a URL is inserted into a community database, the system creates a record that has three fields:

Key: The URL that the user wants to bookmark.

Attributes: A set of attributes that is assigned to the URLs. Attributes are selected from a fixed set known to the community and maintained by the server. Each user may suggest additional attributes, and the server administrator decides whether or not the update will be submitted to the community. The display of bookmarks into folders is done according to their attributes, normally by applying a condition to the attributes of the folder and the attributes of the bookmark. For example, a bookmark is displayed in a folder if they share a common attribute. It is therefore desirable that the community understands the attributes (which means that the bookmark sharing technique described herein will generally be more suitable for small to medium communities). In one preferred embodiment, there are three types of attributes: ***Global*** - an attribute that is shared throughout the community; ***Private*** - an attribute that is private and used only by the user that inserted it. These attributes will not be submitted to other users in the community; and ***Submit*** - an attribute that a user feels should be added to the collection of attributes globally shared by the community. Initially, the attribute is local, but, as will be explained in more detail below, the server can later change them into global attributes.

URL related data: Additional information describing the URL. The related data may be generated automatically. Automatically generated data includes e.g. the bookmark title, time of insertion, importance, owner of the data, expiration date, etc.

After having assigned attributes to both the bookmarks and the folders, the bookmarks can be displayed. Bookmarks are displayed through a user interface, organized for example into a tree of folders such as illustrated in Fig. 1. Each folder has a name and is assigned one or more attributes (taken from the set of attributes). As described above, attributes are also assigned to each bookmark.

In the preferred embodiment, if the folder contains a sub-folder, and a bookmark has also the attributes of the sub-folder, the bookmark is only displayed in the sub-folder (as a finer categorization). This logic is implemented by the following rule:

A bookmark will appear in a folder if and only if:

(a) The bookmark's attributes match the folder.

(b) The bookmark does not appear in any folder contained in this folder.

Thus a bookmark is displayed in a folder if the above condition is met, namely that at least one attribute assigned to the bookmark is contained in the attributes of the folder, and with a further sub-condition stipulating that the bookmark does not appear in any folder contained in this folder. It will be appreciated of course that a variety of conditions (including one or more sub-conditions) can be adopted, depending on the particular circumstances.

As specified above, a user interface is provided for enabling the dynamic assignment of attributes to objects. In accordance with a preferred embodiment, using the interface of Fig. 1, a drag and drop operation is used in order to insert/delete a bookmark. Thus, by dragging a URL into a folder, the URL inherits the attributes of that folder. The operation is simple and intuitive.

Notice that such a single drag and drop of a URL into a folder may result in mapping the URL into multiple folders having a subset of these attributes. The URL may be dragged to additional folders, thereby being assigned with more attributes as necessary. For example, dragging a bookmark to the folder *books* (12) in the user interface illustrated in Fig. 1A, will automatically assign the attributes *shopping* and *book* (13) that are associated with this folder. The side effect of this operation is that this bookmark will also be mapped to the folder *books* (15) in user interface (14), since the folder *books* (15) is also associated with attributes *shopping* and *book* (16). A user may also assign bookmark related data to a URL, such as level of importance, expiration date, etc.

The consequence of a single drag and drop operation is further exemplified with reference to Fig. 3. Fig. 3A illustrates five user views (31 to 35). Having dragged and dropped *item 1* - say, a bookmark (36 in Fig. 3B) to View 5, *item 1* is assigned attributes 1,2 and 4. Consequently, *item 1* is automatically assigned to view 1 (31) -due to attributes 1 and 2, view 3 (33) -due to attribute 4, and view 4 (34) -due to attribute 1 (see Figs. 3C and 3D). The update among the views is implemented in a manner that is described in more detail below.

All users define their own individual tree of folders via the user interface. As mentioned above, the tree defines the folders used by the user to organize and subsequently display bookmarks. Users may edit their tree at any time. These operations are done on the user's machine and no one (i.e. other users or the server) can see the user tree view. In accordance with a preferred embodiment, a server maintains the database of attributes and bookmarks; each member of the community runs a user application. The users are allowed to work off-line (i.e. a client can work even if the server is not running). In order to do so, the client must have a local replica of the data collected by the server. This is done using a data replication scheme between the client and the server. The replication protocol includes, in one preferred embodiment, the following steps:

Initialization: Upon initialization, the user registers with the server and receives the current database and the community attribute list. The database may be organized using any known technique, e.g. as a relational database.

Updates: From time to time, at intervals determined by its internal clock, the client sends the server updates and receives updated information that was sent to the server by other users.

In accordance with a preferred embodiment, the update protocol concerns preferably only incremental updates, and therefore keeps communication overhead to a minimum. Because the user starts with the same database as the server and all client updates are done in the order specified by the server, both the user and server database contain the same information after the update process. This is the reason why incremental updates are sufficient. Note that if two users update the same bookmark, the updated information is determined by the last user to perform an update.

Figs. 4a-c illustrate an update sequence in accordance with one embodiment of the invention. Thus, Fig. 4a illustrates three user replicas **(41-43)** interlinked to a database server **(44)**. Each replica includes objects that are assigned attributes. The replica may further include the user set of attributes and also the user set of containers.

Considering, for example, that a user updates his replica with an additional object (**45** in Fig. 4B) which he wishes to submit to other users, the update (in this particular example, addition of an object) is submitted (**46** and **47** in Fig. 4C) to the user replicas (**42** and **43**) through the server (**44**). The update is received in each replica, and when these users invoke the user interface for displaying objects in containers, the specified update is displayed, if applicable. Thus if the attribute(s) of the specified object and those of container(s) satisfy the relevant condition(s), such as the example given above, the object is displayed in the container(s).

When a community shares information, it is important to maintain user privacy. Most users in a community, although they would like to share information, need to keep certain data private. The preferred embodiment therefore supports a form of privacy. Note that since the server stores all of the community data, certain information must therefore be kept private from the server as well. Users can decide whether or not information will be shared or kept private.

Clients choose a password to protect their private data. The application then selects a random key (client key), and encrypts it, using, say the DES encryption algorithm with the password as the secret key. The client then saves the encrypted key and uses it for login purposes. The system uses the clients key for encryption, and for setting up a shared key with the server, used for authentication.

When a user specifies a bookmark as private, the application encrypts the data using, e.g. DES with the clients key. In this way, the server receives encrypted data; even the server cannot open this data nor can this data be shared with other members of the community. Note that when a client receives an encrypted bookmark, it uses its key to decrypt the data.

Since a single user may install the client on several machines, there must be a way for users to retrieve all their bookmark data (including private items). Hence, a user may export his password to a special encrypted password file. Whenever the user wants to install another client, he will be prompted to import the password file containing the encrypted key. The new client can retrieve the key provided that the same client password is used.

Turning now to Fig. 2, there is shown a block diagram of a generalized system architecture in accordance with one preferred embodiment of the invention. As shown, the system (20) has three main components:

(a) Server (21) that maintains the community bookmarks and manages the system;

(b) Users (Clients) (of which only 22, 23 and 24 are shown in Fig. 2, but typically there will be many more); and

(c) Web browsers $22^{(i)}$, $23^{(i)}$ and $24^{(i)}$ communicating with the respective clients (users).

Communication between the client and the server is performed using TCP/IP. Communication inside the client's machine, between the (possibly multiple) browsers and the client, is done using the DDE protocol (see below). (It will be appreciated of course the other implementations may use different protocols). The DDE protocol connects the client (e.g. 22) with the web browsers on that client (22^1 and 22^2). Most web browsers implement the DDE (Dynamic Data Exchange) protocol for Windows (the DDE service name for Netscape is *NETSCAPE*, and for Microsoft is *EXPLORER*). The client uses known DDE methods to exchange data with the browser. The client records the URLs visited in each browser windows, even if there are several browser windows open. The client will open a URL, specified by the user, in the browser window that was last opened (the active window). This implementation provides the system with an easy way to manipulate bookmarks. There is no need to "cut" and "paste" a URL in order to insert a new bookmark.

The server (21) and client use TCP/IP to communicate with each other. The server performs the administrative tasks required to manage the bookmarks, global attributes, and server preferences. The server's user interface is quite simple.

The server, as mentioned above, manages the system's collection of attributes. The attribute dialog box (user interface) shown in Fig. 5 displays all of the system's current attributes and includes user interface elements that enable the following operations:

Add: (51) adds a new global attribute

Rename: (52) renames a global attribute

Delete: (53) deletes a global attribute

Global: (54) modifies a local (private) attribute so that it becomes a global one

Join: (55) takes either global or local attributes and combines them into one global attribute.

The field title "name" (56) stands for the attribute name, and "own" (57) stands for the attribute owner. The symbols (G) e.g. (58), (L) e.g. (59), and (S) e.g. (59') stand for Global, Local and Submit, respectively. "Global" attributes, as specified above, are shared among all the community

users owned by the server (root), whereas "Local" and "Submit" attributes are owned by the corresponding client (user).

The user interface described with reference to Fig. 5 may be utilized by each user in the community for updating attributes of his/her own set. The updated attributes may then be submitted in the (periodic) update to selected (or all) users in the community, e.g. in accordance with the protocol described schematically with reference to Fig. 4.

Turning now to the client, the user interface (shown in Figure 6) is used to access bookmarks and add bookmarks to the community's collection. Its appearance resembles that of Windows Explorer. The client also enables the user to perform tasks such as editing the tree, assigning attributes to folders or bookmarks, opening a bookmark in the active browser window, and so forth.

In the embodiment shown in Figure 6, the client user interface has the following items:

Menu bar (61) - includes options for File, Browser, Update Attributes and Help.

Left window (62) - displays the folder hierarchy or "tree". Users can click a folder's name to see its contents displayed in the right window.

Right window (63) - displays the contents (either bookmarks or sub-folders) of the folder selected in the left window. Folder contents can be sorted according to Title, Rating, URL, Date or Owner.

Status bar (64) - runs along the bottom of the program window and displays the attributes of the bookmark or folder that is currently selected.

In the preferred embodiment, there is a pre-defined folder - Root (65). This is the root of the folder hierarchy or "tree" and is entitled "*username's* Bookmarks". Any new folder created is displayed beneath this one. Certain standard folders are also pre-defined for the user: eg *Hot* (66) and *History* (67).

Fig. 6 shows the situation where the highlighted URL (entitled "the time zone page" (68)) is being inserted into the highlighted folder *find* (69), and is assigned the attributes "find" and "travel" (64), as shown in the status bar. The condition that is applied to the bookmark and the folder is as discussed above, i.e. a bookmark is displayed if its attributes are included in those of the folder and it does not appear in a sub-folder of the folder.

In the preferred embodiment the user may create folders having the following attributes:

Public or *shared* - These are folders that contain any type of bookmark.

Private - These folders include the "Private" attribute which ensures that a URL dragged into this folder will be marked "private" by default, and kept encrypted in the database.

Separator - A folder without attributes, i.e. the attribute NULL is associated thereto. This is useful to collect several folders which have a common utility but no common attributes, for example, to collect multiple projects under a single "projects" separator, or to collect several cities under "travel".

Folders that do not yet contain attributes (and therefore do not contain bookmarks).

The preferred embodiment supports two types of bookmark:
Public bookmarks - shared with the rest of the community that is connected to the server.

Private bookmarks - specified as private and can only be viewed from that owner's client.

To set the attributes of a bookmark, users can typically do one of the following:

(a) Drag the bookmark into a folder. The bookmark automatically inherits the folder's attributes.

(b) Select a URL (bookmark) so that its name is highlighted and then set its attributes using the URL Attributes dialog user interface as is displayed in Figure 7.

Other bookmark information can be added as follows:

Add a Title and Description to the URL (71): Users are not required to fill in this information, but this description will help the community use the bookmark more efficiently.

Set the importance rating for a bookmark (72): Low indicates a less important bookmark and High indicates a more useful or important site.

Set the bookmark to be automatically deleted after a specified period of time. (73)

Mark a bookmark as Private to prevent it from being shared with other users. (74)

In the particular example of Figure the attributes (75) that were assigned to the URL are *Book* and *Shopping*.

Whilst the example above has focused on shared applications where information (bookmarks) is shared among users in a community, it will be appreciated that similar techniques can be used in non-sharing situation. Thus, for example, a stand-alone mode may be provided where objects are displayed in containers, through a user interface, if for example the object and the container share at least one common attribute. It will also be appreciated that whilst the description has dealt mainly with the addition of attributes to objects, other standard operations such as deletion of attributes are also supported.

Moreover, whilst, the above discussion has concentrated primarily on bookmark management and sharing applications, the invention is by no means bound to bookmarks, but rather can be applied to a wide range of other objects, such as emails, files of various types, etc, or combinations thereof. Likewise, the description has mainly focused on providing a tree of folders for the user interface, but the skilled person will appreciate that folders are only one example of a set of containers, and a tree is only one example of arranging the set of containers. It will further be appreciated that although a user is generally associated above with a single machine, a given user may have two distinct user applications running on the same or different machines.

CLAIMS

1. A method for managing objects for at least one user, comprising:

providing a set of attributes;

providing a set of containers, each associated with one or more attributes from said set;

providing a user interface for dynamically assigning attributes from said set to said objects; and

selectively displaying, through a user interface, at least one container, wherein an object is displayed in said container if a predetermined condition is met, said condition involving at least one attribute of the container and at least one attribute of the object.

2. The method of Claim 1, wherein said dynamically assigning includes mapping an object to at least one container and inheriting by the object the attributes of said at least one container.

3. The method of Claim 2, wherein said mapping includes drag-and-dropping said object to said at least one container.

4. The method of any preceding Claim, wherein at least one container is associated with one or more essential attributes, said condition including the requirement that the attributes of said objects contain said one or more essential attributes.

5. The method of any preceding Claim, wherein there is a community of users that share said objects, each user being associated with a respective set of attributes wherein at least one attribute is common to at least two of said users; the method further comprising the steps, for each user in the community, of:

providing a user replica that includes objects that are each assigned one or more attributes;

providing a set of containers each associated with one or more attributes from said set;

providing a user interface for generating an update to said replica;

submitting the update stipulated to the replicas of selected other users;

receiving at least one update from another user in the community;

updating said user replica in accordance with the at least one received update; and

selectively displaying, through a user interface, at least one container, wherein an object from the replica is displayed in said container if a predetermined condition is met, said condition involving at

least one attribute of the container and at least one attribute of the object.

6. The method of Claim 5, further comprising the step of:
categorizing selected objects as private, and encrypting said objects with a user unique key.

7. The method of Claim 5 or 6, wherein one or more selected containers in at least one user replica are assigned a *private* attribute, whereby any object assigned to such a container is encrypted using a unique key.

8. The method of any of Claims 5 to 7, wherein the set of attributes associated with each user forms part of the user replica.

9. The method of any of Claims 5 to 8, wherein an update includes assigning attributes to said objects.

10. The method of any of Claims 5 to 9, wherein an update includes updating at least one attribute in said set, said updating of an attribute including: adding an attribute, deleting an attribute, editing an attribute, change the status of attribute from Global to Local or *vice versa*, and joining two or more attributes into a single attribute.

11. The method of any preceding Claim, wherein said user interface includes a tree of containers.

12. The method of Claim 11, wherein said condition further includes the requirement that an object is displayed in a container only if it is not displayed in a sub-container thereof.

13. The method of any preceding Claim, wherein said containers are folders.

14. The method of any preceding Claim, wherein said objects comprise URL bookmarks of web sites.

15. The method of any of Claims 1 to 13, wherein said objects comprise files.

16. The method of any preceding Claim, wherein the user interface supports: adding at least one attribute, deleting at least one attribute and updating at least one attribute.

17. A computer program comprising a set of program instructions which when followed on one or more computer systems cause said one or more computer systems to perform the method of any preceding Claim.

18. A computer system for managing objects for at least one user, comprising:

means for providing a set of attributes;

means for providing a set of containers, each associated with one or more attributes from said set;

means for providing a user interface for dynamically assigning attributes from said set to said objects; and

means for selectively displaying, through a user interface, at least one container, wherein an object is displayed in said container if a predetermined condition is met, said condition involving at least one attribute of the container and at least one attribute of the object.

19. The system of Claim 18, wherein said dynamically assigning includes mapping an object to at least one container and inheriting by the object the attributes of said at least one container.

20. The system of Claim 19, wherein said mapping includes drag-and-dropping said object to said at least one container.

21. The system of any of Claims 18 to 20, wherein at least one container is associated one or more essential attributes, said condition including the requirement that the attributes of said objects contain said one or more essential attributes.

22. The system of any of Claims 18 to 21, wherein there is a community of users that share said objects, each user being associated with a respective set of attributes wherein at least one attribute is common to at least two of said users, the system further comprising a processor for each user and a server including:

means for providing a user replica that includes objects that are each assigned one or more attributes;

means for providing a set of containers each associated with one or more attributes from said set;

means for providing a user interface for generating an update to said replica;

means for submitting via the server the update stipulated to the replicas of selected other users;

means for receiving from the server at least one update from another user in the community;

means for updating said user replica in accordance with the at least one received update; and

means for selectively displaying, through a user interface, at least one container, wherein an object from the replica is displayed in said container if a predetermined condition is met, said condition involving at least one attribute of the container and at least one attribute of the object.

23. The system of Claim 23, further comprising means for categorizing selected objects as private, and encrypting said objects with a user unique key.

24. The system of Claim 22 or 23, wherein one or more selected containers in at least one user replica are assigned a *private* attribute, whereby any object assigned to such a container is encrypted using a unique key.

25. The system of any of Claims 22 to 24, wherein the set of attributes associated with each user forms part of the user replica.

26. The system of any of Claims 22 to 25, wherein an update includes assigning attributes to said objects.

27. The system of any of Claims 22 to 26, wherein an update includes updating at least one attribute in said set, said updating of an attribute including: adding an attribute, deleting an attribute, editing an attribute, change the status of attribute from Global to Local or *vice versa*, and joining two or more attributes into a single attribute.

28. The system of any of Claims 18 to 27, wherein said user interface includes a tree of containers.

29. The system of Claim 28, wherein said condition further includes the requirement that an object is displayed in a container only if it is not displayed in a sub-container thereof.

30. The system of any of Claims 18 to 29, wherein said containers are folders.

31. The system of any of Claims 18 to 30, wherein said objects comprise URL bookmarks of web sites.

32. The system of any of Claims 18 to 30, wherein said objects comprise files.

33. The system of any of Claims 18 to 32, wherein the user interface supports: adding at least one attribute, deleting at least one attribute and updating at least one attribute.

The Patent Office
PATENTS • DESIGNS • COPYRIGHT • TRADE MARKS
INVESTOR IN PEOPLE

Application No: GB 0104912.1
Claims searched: All
Examiner: Michael R. Wendt
Date of search: 29 January 2002

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.T): G4A (AKS, AUDB)

Int Cl (Ed.7): G06F3/00; G06F17/30

Other: Online: EPODOC, WPI, Japio

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
P,A	WO 00/55741 A (BLINK.COM) e.g. see page 5 line 15 - page 6 line 20.	

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.